SISTEMA DE RECONOCIMIENTO DE VOCABULARIO AMPLIADO EMPLEANDO RASPBERRY APLICADO A LA DOMÓTICA.

David Ernesto Duque Osorio
Andrés Felipe Marín Zapata

Proyecto de grado presentado como requisito parcial para optar por el título de Ingeniero Electricista
Director:

Julián David Echeverry Correa, PhD.

Universidad Tecnológica de Pereira
Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación
Programa de Ingeniería Eléctrica
Pereira
2018

Contenido

Pág.

1. Resumen

Este proyecto presenta el diseño de un prototipo de bajo costo de un sistema para el control de un entorno domótico, basado en la API de reconocimiento de voz de Google (Google Voice API[1]) y el sistema electrónico embebido Raspberry Pi.

Este proyecto se inició con la configuración de la Raspberry Pi, que incluye tanto la descarga del sistema operativo, como la adjudicación de permisos e instalación de paquetes necesarios para convertir a la Raspberry Pi en un servidor web. El siguiente paso fue la programación de la página web en código HTML, que se enlazaría con el motor de reconocimiento de voz de Google.

A continuación, se eligen y programan en Python las acciones que reconocerá la página web y enviará a la Raspberry, se programa un archivo en PHP que será el encargado de hacer que la Raspberry Pi ejecute las acciones programadas. Todo este conjunto de archivos se guarda en la ubicación correspondiente al servidor web en la Raspberry.

Debido a un cambio en los protocolos de seguridad del navegador Google Chrome, el cual es el único navegador donde funciona correctamente el reconocimiento de voz de Google, se hizo necesario la utilización del software Wampserver que entre muchas otras características permite que estos cambios en los protocolos de seguridad de Google Chrome no impidan el correcto funcionamiento del sistema de reconocimiento de voz en la página web.

Por último, se diseñó y conectó a la Raspberry el circuito electrónico con que el que se hizo posible la visualización del correcto funcionamiento del prototipo.

2. Introducción

El hombre siempre ha buscado simplificar y facilitar la realización de tareas, lo que lo ha llevado a realizar grandes avances en el desarrollo tecnológico como la invención de máquinas para ayudar a facilitar y mejorar la vida y los procesos productivos. Esta aparición de las máquinas generó un nuevo tipo de comunicación entre el hombre y la máquina.

La comunicación hombre-máquina ha ido cambiando a través del tiempo buscando que cada vez sea de manera más eficiente y sencilla, esta comunicación se ha dado por lo general por medio de un contacto físico (por medio de botones, palancas, teclados, ratones, pantallas táctiles, entre otros).

El siguiente paso en la comunicación hombre-máquina era lograr una interacción con las máquinas por medio de la voz ya que es una manera más natural y que permitiría a las personas mayores y con ciertas discapacidades físicas ser mucho más independientes y productivas, además de mejorar la calidad de vida de todas las personas.

[1] https://cloud.google.com/speech/?hl=es

Con el desarrollo de la electrónica y su masificación surgieron nuevas necesidades como la de automatizar algunos procesos, lo que introdujo el término edificios inteligentes y más adelanta aparecieron los entornos domóticos (mejor conocido como casas inteligentes) lo que permitió la realización de algunas tareas por medio de un control remoto.

3. Preliminares

3.1. Planteamiento del problema

En la actualidad la mayoría de los sistemas domóticos son operados por dispositivos táctiles o mandos de botones. Muy pocos son los sistemas domóticos controlados por la voz, que se ofrecen en el mercado. Además, es difícil encontrar entre los sistemas domóticos comerciales que se ofrecen actualmente, uno que permita cualquier tipo de modificación. [12]

Realizar el control de un sistema por medio de la voz resulta ser una manera rápida, intuitiva y natural, también es un método que libera la visión y las manos al usuario. A diferencia de los métodos convencionales que requieren que el usuario preste una mayor atención al dispositivo de control.

En ocasiones se requiere que los dispositivos de un hogar sean activados y desactivados por medio de la voz. La necesidad de aislamiento y la comodidad son algunas de estas razones. Para satisfacer estas necesidades se puede aplicar un sistema de reconocimiento de voz, el cual debe ser económico, eficiente, simple y de bajo consumo de energía.
Un sistema de reconocimiento de voz debe poder programarse de una manera sencilla y que además sea veloz, por lo que se requiere que el elemento en el cual se programa sea poco robusto y rápido.

El sistema debería permitir varios comandos de voz para la ejecución de una acción, esto debido a que la orden emitida por el usuario debe contener la acción y el lugar donde se desea llevar a cabo, como por ejemplo “abrir puerta garaje”.

Como el usuario está limitado a pronunciar una frase específica para dar una orden, la programación de las frases debe ser lo más sencilla de deducir.
El desarrollo resultante podría ser un código abierto que permita su modificación y estudio posterior, para que se siga perfeccionando el sistema o por si se desea hacer compatibles otros desarrollos con este sistema.

3.2. Justificación

Por los continuos avances tecnológicos en los últimos años, el deseo de automatizar nuestro entorno ha tomado un papel importante, como por ejemplo la comunicación hombre máquina, la cual podría tener una gran mejora al integrársele sistemas de

reconocimiento de voz cada vez más robustos y confiables. Para el ser humano la comunicación por medio del habla es más natural y sencilla, por lo que emplear el reconocimiento de voz para el control de un sistema es una herramienta útil. El que solo se dependa del habla para realizar algunas tareas es de gran ayuda para personas con ciertas discapacidades físicas. Además, por medio del reconocimiento de voz se pueden eliminar algunas limitaciones de la actual interacción hombre máquina: se podría controlar un sistema estando en la oscuridad o sin estar frente a un teclado. Este método permite a cualquier persona con capacidad de hablar controlar algunos dispositivos del hogar mediante la voz.

Algunos sistemas electrónicos embebidos son dispositivos económicos y versátiles que permiten ser utilizados para la programación de operaciones de forma sencilla. También se pueden integrar al motor de reconocimiento de voz Google, que es una herramienta en continua evolución, los vuelve una excelente opción para utilizarse en el desarrollo de sistemas de control por medio del habla.

Dentro del perfil ocupacional del Ingeniero Electricista de la Universidad Tecnológica de Pereira, se encuentra el diseño e implementación de software y hardware mediante las nuevas tecnologías para la sistematización y control de procesos.

3.3. Objetivos

3.3.1. Objetivo general

Diseñar un sistema de reconocimiento de vocabulario ampliado empleando Raspberry aplicado al control de un entorno domótico.

3.3.2. Objetivos específicos

- Estudiar los sistemas de reconocimiento automático de habla y las interfaces con dispositivos embebidos.
- Implementar un sistema automático de reconocimiento de habla con vocabulario ampliado y aplicado a la domótica.
- Validar el sistema de reconocimiento en la utilización del mismo para el control domótico empleando métricas de evaluación como el WER y el Accuracy.
- Diseñar un dispositivo que permita controlar un número de elementos mayor al número de salidas de la Raspberry Pi.

4. Marco teórico

En esta sección se mencionan todos los recursos usados para la realización del proyecto, incluyendo una breve descripción de cada uno de ellos.

4.1. Hardware

A continuación, se hablará definirán algunos elementos de la Raspberry pi y de los elementos que constituyen el circuito electrónico que está encargado de aumentar las salidas de la Raspberry pi.

El sistema embebido Raspberry Pi es un computador de placa reducida (SBC), utiliza un procesador diferente al de las computadoras convencionales por lo que no es compatible con los sistemas operativos Microsoft Windows y MAC, pero permite instalar versiones del sistema operativo Linux. Este sistema embebido no posee disco duro por lo que es necesario utilizar una tarjeta de memoria SD, posee puertos USB para conectar periféricos como el ratón, teclado o un receptor de red Wifi. Diseñado en el Reino Unido por la fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Raspberry Pi cuenta actualmente con cinco modelos diferentes: Raspberry Zero, Raspberry 1 modelos A+ y B+, basados en Raspberry Modelos Ay B, Raspberry Pi 2 modelo B, Raspberry Pi 3 Modelo B.

Raspberry pi 1 modelo b

En este documento se describirá principalmente la Raspberry Pi 1 modelo B, que se muestra en la figura 1, ya que fue la utilizada en el desarrollo del proyecto.

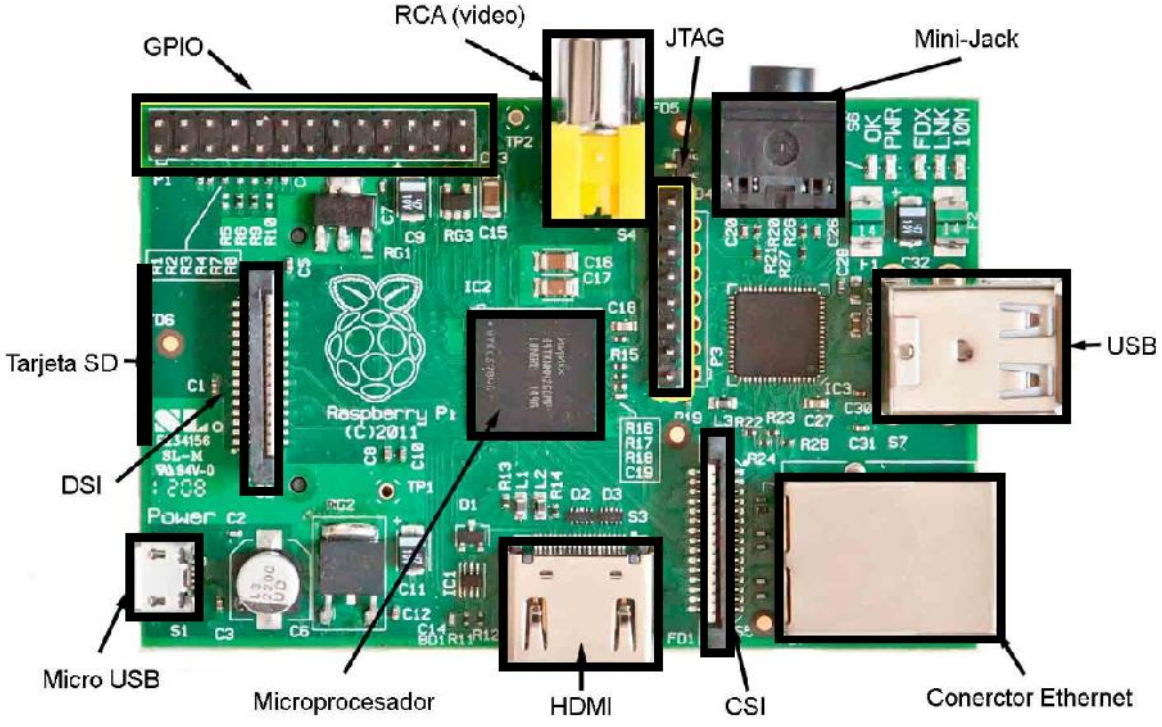

Figura 1. Componentes Raspberry Pi 1 modelo B.

Puertos USB

Este modelo posee dos puertos USB, permitiendo conectar teclado y ratón, dispositivos de almacenamiento externo USB, Wifi USB, entre otros.

Puerto ethernet

Incluye puerto Ethernet para conecta la Raspberry a red cableada, lo que permite tener un acceso a Internet y hace posible que otros dispositivos en la red puedan acceder a la Raspberry Pi.

Conector CSI (*camera serial interface*)

Es un conector tipo bus de 15 pines utilizado para añadir un dispositivo compatible con la interfaz CSI-2 (Camera Serial Interface versión 2). Donde es posible conectar la cámara de la Raspberry Pi.

Conexión a pantalla

Raspberry Pi es compatible con tres salidas de video diferentes: video compuesto, video HDMI y video DSI. El video DSI requiere de un hardware avanzado para poder ser utilizado, mientras que el video compuesto y video HDMI son más accesibles para el usuario.

Conexión de audio

No es una conexión necesaria si se conecta una pantalla por medio del puerto HDMI ya que este es capaz de transportar las dos señales tanto la de video como la de audio, si se está usando la salida de video compuesto la Raspberry solo estaría transportando video por lo que si se desea la señal de audio analógica estará disponible una ranura de audio Jack de 3.5 mm.

Almacenamiento

Raspberry Pi viene por diseño sin disco duro por lo que cuenta con una ranura para memorias SD, casi cualquier tarjeta SD funcionará con Raspberry Pi. Está tarjeta SD debe ser de al menos 2 GB ya que será la que albergará todos los archivos requeridos por el sistema operativo.

Alimentación

Raspberry Pi se enciende automáticamente con la conexión a la fuente de alimentación por lo que no posee botones de encendido y apagado. Para su alimentación se dispone de un conector micro USB que suministra 5 Voltios y mínimo 0.7 Amperios.

Pines GPIO (*general inputs and outputs*)

El diseño de los pines del puerto GPIO de Raspberry Pi varía ligeramente según el modelo de placa que se posea. Todos los modelos actuales más recientes cuentan con el mismo diseño de pines. Antes de utilizar el puerto GPIO, se debe conocer el modelo de Raspberry Pi que se está utilizando.

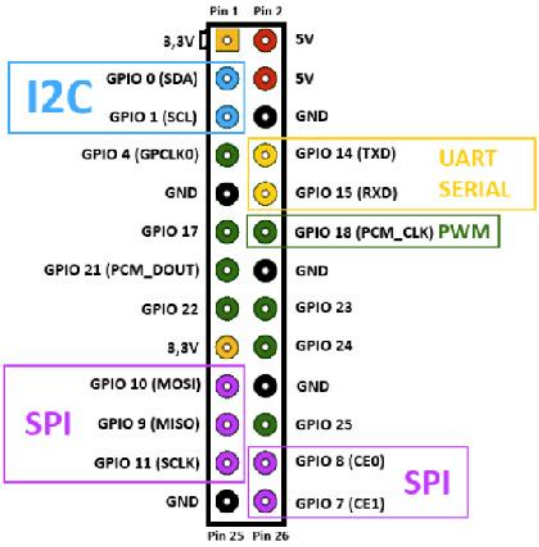

Figura 2. Diagrama puerto GPIO Raspberry Pi 1 modelo B.

Para aumentar el número de salidas del puerto GPIO y así aumentar el número de dispositivos controlados, se utiliza un circuito electrónico el cual consta de un decodificador y de flip flops.

Decodificador

El decodificador es un dispositivo que acepta una entrada digital codificada en binario y activa una salida. Este dispositivo tiene varias salidas, y se activará aquella que establezca el código aplicado a la entrada.

Flip flop J-K

El flip flop es el nombre común que se le da a los dispositivos de dos estados (biestables), que sirven como memoria básica para las operaciones de lógica secuencial. Los flip flops son ampliamente usados para el almacenamiento y transferencia de datos digitales y se usan normalmente en unidades llamadas “registros”, para el almacenamiento de datos numéricos binarios. Solo toma dos posibles valores en sus salidas q según la tabla de verdad.

$\overline{\text{Preset}}$	$\overline{\text{Clear}}$	CK	J	K	Q	$\overline{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0			NA		
1	1	↓	0	0	Q_0	$\overline{Q}_0$
1	1	↓	1	0	1	0
1	1	↓	0	1	0	1
1	1	↓	1	1	$\overline{Q}_0$	Q_0
1	1	0, 1	X	X	Q_0	Q_0

Figura 3. Tabla de verdad de flip flop J-K.

4.2. Software

En esta sección se mencionará todo lo respectivo al software y lenguaje de programación que hicieron posible la realización del proyecto.

Sistema de reconocimiento de voz

Un sistema de reconocimiento de voz es una herramienta computacional capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. En este proyecto se usará el reconocimiento de voz en conjunto con una aplicación web, que será la encargada de enviar al dispositivo elegido para ejecutar las acciones, lo reconocido por el sistema de reconocimiento de voz. [13]

Para que un sistema de reconocimiento de voz sea eficiente debe tener en cuenta los siguientes aspectos:

- El usuario puede hablar de manera continua o con palabras aisladas.
- Cada usuario tiene un tono de voz y pronunciación diferente.
- Diversidad de idiomas.
- Ruido externo.

El sistema de reconocimiento de voz utilizado en este proyecto es el de la aplicación SPEECH de Google, ya que Aplica los algoritmos más avanzados de redes neuronales del aprendizaje profundo al audio de los usuarios para conseguir un reconocimiento de voz muy preciso. La precisión de la API Speech mejora con el tiempo, conforme se perfecciona la tecnología interna de reconocimiento de voz. SPEECH también puede transmitir resultados de texto conforme vaya reconociendo el audio, de forma que el texto reconocido aparece inmediatamente mientras la persona habla. De igual modo, la API puede reconocer el texto a partir del audio almacenado en un archivo. [14]

Sistema Operativo Raspberry Pi

Se necesita instalar el sistema operativo en la Raspberry para poder descargar el servidor Apache y que este pueda funcionar correctamente. El sistema operativo se instala descargando una imagen del sistema operativo desde la página de Raspberry Pi[2] y posteriormente se carga a la tarjeta SD. Para este proyecto se usó la distribución RASPBIAN JESSIE.

Linux

Linux es un sistema operativo de código abierto lo que quiere decir que se puede descargar el código fuente del sistema operativo y hacer los cambios que se deseen. Esta

[2] http://www.raspberrypi.org/downloads

forma de desarrollo es lo que ha permitido a Linux ser modificado para ejecutarse en la Raspberry Pi.

Debian

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo (SO) libre. Los sistemas Debian actualmente usan el núcleo de Linux o de FreeBSD. Linux es una pieza de software creada en un principio por Linux Torvalds y desarrollada por miles de programadores a lo largo del mundo.

Servidor web

En este proyecto la Raspberry PI funciono como servidor web, esperando las órdenes enviadas desde la aplicación web para posteriormente ejecutarlas.

Un servidor web que se ejecuta en un ordenador está a la espera de solicitudes por parte del usuario, cuando este recibe una solicitud responde enviando el código en HTML de la página y a continuación este código enviado por el servidor es interpretado y mostrado en la pantalla del navegador. Esto es necesario ya el servidor solo se encarga del envío del código más no de su interpretación. Además de encargarse del envío de código HTML los servidores web también pueden ejecutar aplicaciones web, estas páginas están formadas por código que se ejecuta cuando se realiza alguna solicitud o respuesta HTTP.

Existen aplicaciones en el lado del usuario y en el lado del servidor. En el primero, el servidor proporciona el código de las aplicaciones y el usuario se encarga de ejecutarlas, en el lado del servidor, el servidor web ejecuta la aplicación y una vez ejecutada genera un código HTML que se devuelve al servidor, el cual envía este código al usuario por medio del protocolo HTTP.

Apache

Apache es en la actualidad uno de los servidores web más populares. Es de código libre, robusta cuya implementación se realiza de forma colaborativa, con presentaciones y funciones equivalentes a la de los servidores comerciales. El proyecto Apache está dirigido y controlado por un grupo de voluntarios de todo el mundo que usan Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada.

Dispone de una gran cantidad de módulos que lo convierten en un servidor capaz de gestionar una gran variedad de aplicaciones, dispone de módulos para:

- Implementar SSL (Protocolo de seguridad en la transferencia de información).
- Enlace con el servidor Tomcat de aplicaciones, para implementar aplicaciones Java de servidor.
- Módulo para PHP
- Módulo para PYTHON.

- Entre otros.

Raspberry Pi como servidor web

En este proyecto la función principal de la Raspberry Pi es como un servidor ya que almacenará las acciones programadas y estará disponible esperando que se le envíen los datos desde una página web. La Raspberry Pi es ideal para actuar como servidor ya que estará en un ambiente liviano, con pocos usuarios. Aunque tiene poco poder de procesamiento y memoria limitada, su bajo consumo de energía y su ejecución silenciosa la convierten en una buena opción para páginas sencillas con bajo tráfico en una red local o en internet.

Gran parte de los servidores web de la actualidad funcionan ejecutando una combinación Linux, Apache, MySQL y PHP (conocida como la pila LAMP). Linux se encarga de proveer el sistema operativo, MySQL el soporte para la base de datos, Apache el servidor web, y PHP el lenguaje de scripting para las páginas dinámicas. Con un servidor basado en LAMP, se hace posible la ejecución de paquetes complejos que van desde sistemas de gestión de contenidos como WordPress hasta foros interactivos.

Diseño de páginas web

En este proyecto usaremos tres lenguajes de programación los cuales son: HTML5 y PHP.

HTML5

Es el lenguaje más utilizado en el desarrollo de páginas web ya que provee características como: estructura, estilo y funcionalidad. Es considerado el producto de la combinación de HTML, CSS y Javascript. Estos tres lenguajes actúan como una unidad organizada bajo la especificación de HTML5. HTML se encarga de la estructura, CSS presenta la estructura y su contenido en la pantalla y Javascript usado para páginas dinámicas.

HTML usa un lenguaje de etiquetas para construir páginas web. Estas etiquetas HTML son palabras clave y atributos rodeados de los signos mayor y menor. La mayoría de las etiquetas HTML se utilizan en pares y que pueden ser anidados (contenidos uno dentro de otro), una etiqueta de apertura y una de cierre y el contenido se declara entre ellas.

PHP

PHP es un lenguaje de programación en el lado del servidor web, el código fuente en este lenguaje es igual que el de HTML puede ser escrito mediante cualquier editor de texto, el código PHP tiene la estructura global del código HTML: etiquetas de cabecera, titulo, cuerpo, etcétera.

Dentro de un documento en lenguaje HTML es posible insertar scripts de PHP que pueden ser llamados por el código fuente en HTML para ejecutar dichos scripts, tales como: mostrar mensajes en la pantalla, escribir mensajes, ejecutar otros scripts, entre otros. Por

este motivo se hace necesario el uso de scripts en PHP, ya que estos tienen la función para ejecutar el script de Python para controlar el puerto GPIO de Raspberry Pi.

PYTHON

Es un lenguaje de programación cuya sintaxis es muy simple, clara y sencilla. Otra de las razones para usar Python es la compatibilidad con Raspberry Pi, ya que puede interactuar con los pines del puerto GPIO del dispositivo simplemente llamando las funciones codificadas.

Wampserver

Provee características a los desarrolladores con los cuatro elementos necesarios para un servidor web: un sistema operativo (Windows), manejo de base de datos (MySQL), software para servidor web (Apache) y software de programación script web (PHP, Python o PERL). WAMP incluye, además de las últimas versiones de Apache, PHP y MySQL, versiones anteriores de las mismas, para el caso de que se quiera testear en un entorno de desarrollo particular.

En este proyecto fue muy importante el uso de Wampserver ya que además de las características mencionadas anteriormente, posee una característica adicional la cual es capaz de hacer que una página web, que para Google Chrome es aparentemente no segura, lo sea.

Debido a los nuevos protocolos de seguridad de Google, no es posible tener acceso a la cámara ni al micrófono en las páginas que Google Chrome detecta como no seguras. Por esta razón fue necesario la utilización de Wampserver.

Artyom

Artyom.js es un paquete robusto pero fácil de usar de las API speechSynthesis y webkitSpeech-Recognition. Permite agregar comandos de voz a un sitio web de manera sencilla, con lo que sería posible emular sistemas de reconocimiento de voz como Google Now, Siri, Cortana, entre otros. Artyom también brinda la posibilidad de procesar los comandos con un lenguaje de servidor en lugar de JavaScript, se puede habilitar el modo remoto de Artyom y así usar el método artyom.remoteProcessorService. [8]

5. Estado del arte

En este capítulo resumiremos brevemente los inicios del reconocimiento de voz y de la domótica, así como sus avances tecnológicos a través el tiempo.

5.1. Reconocimiento de voz

A principio de los años 50 los sistemas de reconocimiento de voz solo comprendían dígitos. En el año 1952 los laboratorios Bell diseñan el Audrey, el cual reconoce los dígitos hablados por una sola voz. Diez años después IBM en la feria mundial de 1962 presenta el ShoeBox la cual podía comprender 16 palabras pronunciadas en inglés. [6]

En los años 70 se da uno de los mayores avances en los sistemas de reconocimiento de voz gracias al interés y al financiamiento del departamento de defensa de los Estados Unidos. El programa DARPA fue uno de los más grandes de su tiempo en la historia del reconocimiento de voz, fue el responsable por el sistema de reconocimiento del habla Harpy de Carnegie Mellon. Harpy podía entender 1011 palabras, aproximadamente el vocabulario de una persona de 3 años. [6]

En los años 80 gracias a nuevos enfoques para comprender lo que dice la gente, los vocabularios del reconocimiento de voz pasaron cientos de palabras a varios miles. Una razón importante fue un nuevo método estadístico conocido como el modelo oculto de Markov. Esto sería la base del reconocimiento de voz en las próximas dos décadas. Equipado con estos vocabularios más extensos, el reconocimiento de voz inicia su camino para trabajar en aplicaciones comerciales para los negocios e industrias especializadas. [6]

En los años 90 los equipos con procesadores más rápidos finalmente llegan y el software del reconocimiento de voz se hace viable para la gente común. En 1990 la empresa Dragon lanza el primer producto de reconocimiento de voz de consumo llamado Dragon Dictate, aunque a un precio bastante elevado. Siete años después lanzan una versión mejorada Dragon Naturally Speaking, esta aplicación reconocía el habla continua, así se podía hablar de forma más natural, alrededor de 100 palabras por minuto. En 1996 BellSouth lanza su sistema de reconocimiento de voz interactivo VAL, que daba información en base a lo que una persona decía por teléfono. [6]

En el 2001 el reconocimiento de voz por computador superaba un 80% de precisión, pero al final de la década parecía que los avances tecnológicos estaban estancados. Los sistemas de reconocimiento de voz funcionaban bien con el lenguaje era limitado.

El desarrollo de la tecnología del reconocimiento de voz volvió a tomar importancia con la llegada de la aplicación de búsqueda de Google Voice para iPhone. El impacto de la aplicación de Google se debe a que los teléfonos celulares y otros dispositivos móviles son ideales para el reconocimiento de voz y que también Google tenía la capacidad de descargar le procesamiento de la aplicación a sus centros de datos en la nube para aprovechar todo ese poder de cómputo para hacer las combinaciones entre palabras del usuario y el enorme número de muestras de habla humana que ellos reunieron.

En 2010 Google añadió el reconocimiento personalizado a su búsqueda por voz (Voice Search) a los teléfonos Android, por lo que el software ahora podía grabar la búsqueda por voz de los usuarios y así producir un modelo de voz más preciso. En 2011 fue añadida la búsqueda por voz al navegador Google Chrome. [7]

En el 2016 el ingeniero Leonardo Martínez realiza un trabajo de grado en la facultad de ingenierías de la Universidad Tecnológica de Pereira, donde aprovechando la facilidad y confiabilidad del sistema de reconocimiento de voz de Google y la economía y facilidad de programación del sistema electrónico embebido Raspberry para llevar un cabo un sistema de reconocimiento de voz que funcionaba por wifi desde cualquier dispositivo móvil o computador. [9]

5.2. Domótica

Inicia a comienzos de los años 70, cuando aparecen los primeros dispositivos de automatización en edificios. En los años 80 la domótica consigue integrar el sistema eléctrico y el sistema electrónico gracias a que los sistemas integrados se empezaron a usar a un nivel comercial.

El desarrollo de la informática permitió la expansión del sistema, principalmente en países como Estados Unidos, Alemania y Japón. El auge de la informática facilito la incorporación en los edificios el sistema cableado estructurado que facilita la conexión de terminales y redes, debido a esta incorporación los edificios reciben el nombre de inteligentes por su automatismo al servicio del propietario.

El primer programa utilizado fue el Save creado en Estados Unidos en 1984, permitió lograr eficiencia y bajo consumo de energía en los sistemas de control de edificios inteligentes. Estas instalaciones regían bajo el sistema X-10, protocolo de comunicaciones que opera a través del accionar de un control remoto, fue desarrollado en 1976 por la empresa escocesa Pico Electronics, sigue siendo la tecnología más utilizada al transmitir datos por líneas de baja tensión.

La domótica tuvo un progreso a gran escala con el desarrollo de las redes informáticas de comunicación, ya sea sistema cableado o Wifi, este avance permitió integrar de manera eficiente todos los dispositivos de una casa.

Sistemas de desarrollo como el ZigBee permiten conformar un protocolo inalámbrico de comunicación domótica, al requerir una baja tasa de envío de datos es en la actualidad uno de los protocolos más requeridos para las casas inteligentes, ya sea en sensores de movimiento, detectores de humo y otras funciones de seguridad.

En los últimos años el mercado de ofertas se ha extendido, permitiendo encontrar diversas variantes de equipos domésticos de integración domótica, como es el caso del EIB que es un conductor eléctrico que ayuda a optimizar los distintos sistemas de seguridad y funcionalidad que componen una casa. [4]

6. Desarrollo

Esta sección contiene todo el proceso necesario para poder llevar a cabo este proyecto. Se detalla el proceso de análisis, diseño, implementación y pruebas, de la interfaz, y de un dispositivo capaz de aumentar el número de salidas de la Raspberry Pi. Además, se describe la configuración de la Raspberry Pi y la computadora.

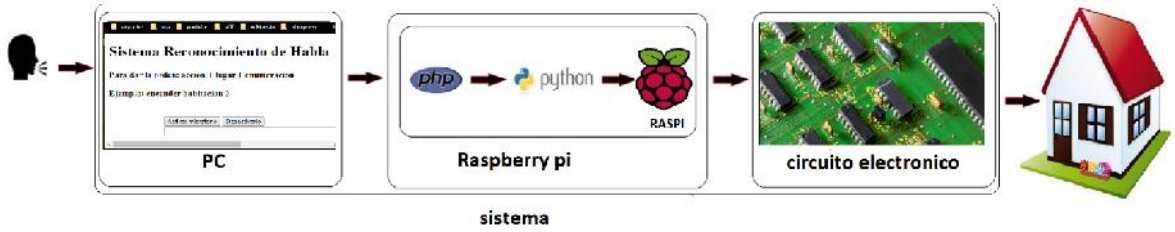

Figura 4. Modelo del sistema domótico controlado por la voz.

En el diagrama de flujo de la figura 4 se observa el proceso realizado en implementación del proyecto. Inicia con el usuario dando comandos de voz a través de un micrófono a un computador. Estos comandos van a una aplicación web, la cual se ejecuta por medio del navegador Chrome de Google. Es fundamental que el navegador que se utilice sea Google Chrome, ya que la aplicación web utiliza para el reconocimiento del habla, el motor de reconocimiento de voz SPEECH de Google. Debido a los protocolos de seguridad de Google, la aplicación web solo puede ser utilizada por un computador que cuente con este navegador y que soporte la aplicación Wampserver.

Una vez que La aplicación web recibe los comandos de voz, los envía al motor de reconocimiento de voz de Google, los cuales retornan en una secuencia de caracteres (palabras en forma de texto) más probable. Ahora, estas cadenas de texto se convierten entonces en los comandos que el sistema debe evaluar para determinar qué acción ejecutar en la simulación del sistema domótico. Si el comando no ha sido programado la aplicación no hace nada, pero si es lo contrario, la aplicación codifica el comando de voz, para facilitar la programación. Después de esto la aplicación web se conecta con una dirección IP correspondiente al servidor montado en la Raspberry. Una vez se establece la conexión, un script de PHP (procesa.php) se encarga de leer el dato enviado por la aplicación web y de ejecutar el script de PYTHON correspondiente a la acción asociada al comando recibido. Además, la aplicación web también se encarga de mostrar en la pantalla del computador el texto que es resultado del proceso de reconocimiento automático de habla. La interacción entre el computador y la Raspberry Pi se puede apreciar en el diagrama de flujo de la figura 5.

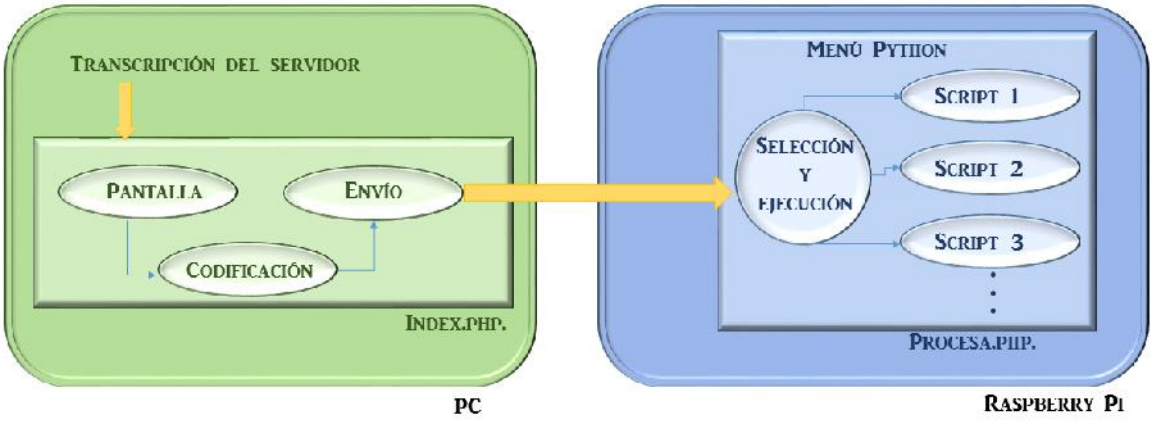

Figura 5.

El script procesa.php es el encargado de recibir los comandos enviados por la aplicación web. Al recibir un comando, lo evalúa y determina que script de Python debe ejecutar. Los scripts de Python están encargados de controlar las salidas del GPIO de la Raspberry Pi. Al ejecutarse un script de Python, la salida en el GPIO activa algunos de sus pines por un momento, tomando el valor de un número en código binario. Lo que hacen los scripts de Python, es controlar los elementos de la simulación del entorno domótico por medio del puerto GPIO.

Hasta la fecha, este proyecto es el único trabajo de reconocimiento de voz empleado para ejercer control sobre un sistema domótico que se ha desarrollado en la Universidad Tecnológica de Pereira. Este proyecto incluye una solución al problema generado por los cambios en los protocolos de seguridad del navegador Google Chrome, que ya no permiten usar el micrófono en proyectos similares desarrollados en esta Universidad. No se ha encontrado literatura relacionada al reconocimiento de voz que aplique el Wampserver como solución a este problema como se ha empleado en este proyecto.

En la sección 6.1. se muestra cómo se configuró la Raspberry Pi para que pudiera ejecutar la acciones que fueran ordenadas desde el computador. Se le dieron permisos y después se le agregaron los scripts de PHP y Python en el lugar adecuado para que apache pudiera controlarlos. También se muestra cómo se configuró el computador para que alojara la aplicación web y que permitiera que esta actuara correctamente.

En la sección 6.2 se muestra cómo se debe hacer la correcta conexión de los elementos del sistema. Como también se muestra el diseño del circuito eléctrico que multiplica las salidas de la Raspberry Pi y que además permite memorizar las acciones.

6.1. Software

En esta sección se describe el proceso de configuración de la computadora y la Raspberry Pi para ser conectadas entre ellas.

La interfaz con el usuario se desarrolló mediante una página web escrita en lenguaje PHP (index.php). En esta página se le pide al usuario que interactúe con la simulación del

entorno domótico mediante comandos de voz. El usuario debe presionar un botón para activar el micrófono del equipo e iniciar la captura de voz. Una vez el usuario haya pronunciado el comando con las instrucciones que quiere ejecutar en el entorno domótico, la aplicación web se encarga de enviar la señal de voz al motor de reconocimiento de Google (Speech). Este motor se encarga de procesar la señal de voz y de retornar la secuencia de caracteres (en forma de una cadena de texto) que ha reconocido a partir de la voz procesada, como se muestra en la figura 6. La aplicación web se encarga de recibir esta cadena de caracteres, procesarla y determinar qué acción tomar. Artyom permite que el reconocimiento de voz inicie cunado el usuario comienza a hablar y termine cuando este haya dejado de hablar por más de 2 segundos. Esto permite que la aplicación web puede procesar frases completas.

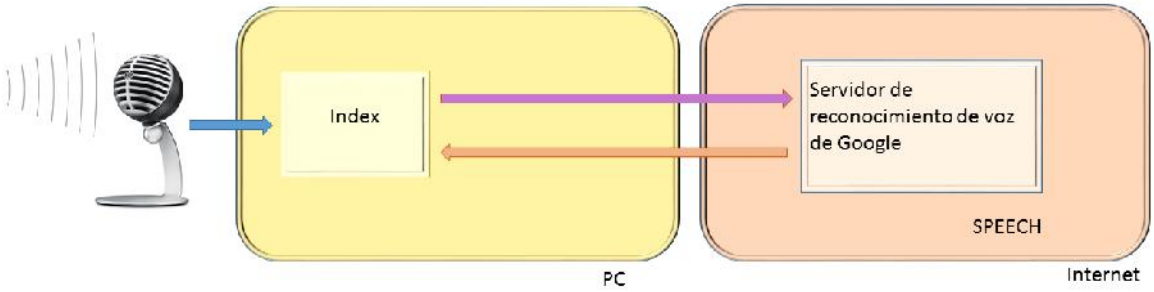

Figura 6.

Luego se inició la configuración de la Raspberry Pi y después se ubicó el archivo PHP en el directorio var/www/html para que pudiera ser ejecutado vía internet. Las acciones que ejecuta el puerto GPIO son programadas en Python y también debieron ser ubicadas en el mismo directorio. Después se configuró el computador, que es el encargado de recibir los comandos de voz, para que la aplicación web pudiera ser utilizada. Se ubicó la aplicación web en la carpeta de instalación del programa Wampserver para que pudiera ser ejecutada con este programa. Así podría utilizar el micrófono sin importar los protocolos de seguridad de Google.

En esta parte del trabajo se debe recordar que los objetivos de este proyecto son simular un entorno domótico y desarrollar un sistema que lo controle mediante una interfaz basada en el reconocimiento de voz. Se debe recordar también que un entorno domótico está compuesto por múltiples elementos y un igual número de actuadores. Por poner un ejemplo sencillo, un apartamento pequeño tiene como mínimo un par de habitaciones, un salón, un espacio para el comedor, una cocina, y uno o varios baños. Cada uno de estos espacios del apartamento tiene, como mínimo, un elemento de iluminación, y un número de dispositivos eléctricos y electrónicos susceptibles de ser controlados por la domótica (ventanas, puertas, aire acondicionado, etc.). En este ejemplo, queda claro entonces que el sistema de control domótico debe ser capaz de controlar de manera simultánea un número de distintos actuadores (uno por cada elemento de los ya mencionados).

Por lo anterior se hizo necesario que el prototipo que se diseñó en este trabajo tuviera un número de salidas que permitiera actuar sobre cada elemento. Hay que anotar que la Raspberry pi posee únicamente 17 salidas digitales, lo que llevó a buscar una solución desde la electrónica digital que permitiera, con un número de tan solo 5 salidas, controlar los 15 elementos que se van a simular. Esta limitación se pudo sortear mediante un circuito decodificador con el cual, a partir de 5 señales de salida de la Raspberry pi, es posible obtener 32 salidas para controlar 16 elementos. Permitiendo contar con 12 salidas más para ser usadas.

6.1.1. Configuración de la Raspberry Pi

En esta sección se presenta cómo se configuró la Raspberry pi para que funcionara como servidor web. Esto para que se le pudiera enviar información desde el computador vía internet y así ejecutar el script procesa.php. Además, se dieron los permisos para que Apache pudiera ejecutar los scripts que controlan los puertos GPIO por medio de procesa.php.

Una vez instalado el sistema operativo el primer paso fue dar el login y la contraseña, los cuales son por defecto “pi” y “raspberry”, respectivamente. Luego se procedió a la configuración de la Raspberry para que pudiera funcionar como un servidor, instalando Apache. Esto es necesario para establecer la conexión entre la Raspberry y el computador vía internet.

Una vez se ingresó al entorno gráfico de Debian se procedió a actualizar los repositorios de la Raspberry. Esto se hace escribiendo el comando `sudo apt-get update` en el terminal. Este comando actualiza la lista de paquetes disponibles y sus versiones, pero no instala o actualiza ningún paquete.

Después de tener los repositorios actualizados, se ingresó el comando `sudo apt-get install apache2 php5 libapache-mod-php5`, que es el que ejecuta el programa apt-get, para que instalara Apache, PHP y libApache2-mod-php5. Este último se encarga de integrar correctamente PHP en Apache.

Cuando se terminó la instalación se reinicia el servidor Apache con el comando `Sudo /etc/init.d/Apache2 restart`, para que las configuraciones hechas por libapache2-mod-php5 surgieran efecto en el servidor.

La instalación de Apache crea el usuario y grupo “www-data” y además el directorio /var/www, que es en donde se debe almacenar el contenido que usa Apache, en este caso los scripts de PHP y Python. Éste usuario, grupo y directorio tienen bloqueados los permisos de ejecución, lectura y escritura por defecto, por lo que deben ser desbloqueados.

Los permisos al usuario y grupo www-data se dan por medio del comando `sudo chown www-data:www-data /var/www`. Y los Permisos de ejecución, lectura y escritura al directorio /var/www/ se dan por medio del comando `Sudo chmod`

`775 /var/www/`. Y por último se incluyó el usuario pi al usuario www-data por medio del comando `Sudo usermod –a –G www-data pi`.

Para darle permisos de administrador al usuario www-data. se usó el comando `sudo visudo`, con el cual se editó el fichero donde se alojan los usuarios con permisos de administrador.

En la figura 7 se puede ver que el usuario Pi tiene permisos de administrador. Se dieron estos permisos al usuario www-data agregando un código igual al del usuario pi, pero cambiando Pi por www-data. Y luego se guardó el archivo con las teclas "ctrl" y "o" al mismo tiempo, y después "ctrl" y "x" de igual manera para salir. Con esto se permitió la ejecución de scripts de Python a través de PHP, cuando se ejecuta el script PHP desde Apache.

Para terminar con la configuración se instalaron las librerías que se requieren para controlar los pines del GPIO de la Raspberry Pi con el comando `sudo apt-get install Python-rpi.GPI`.

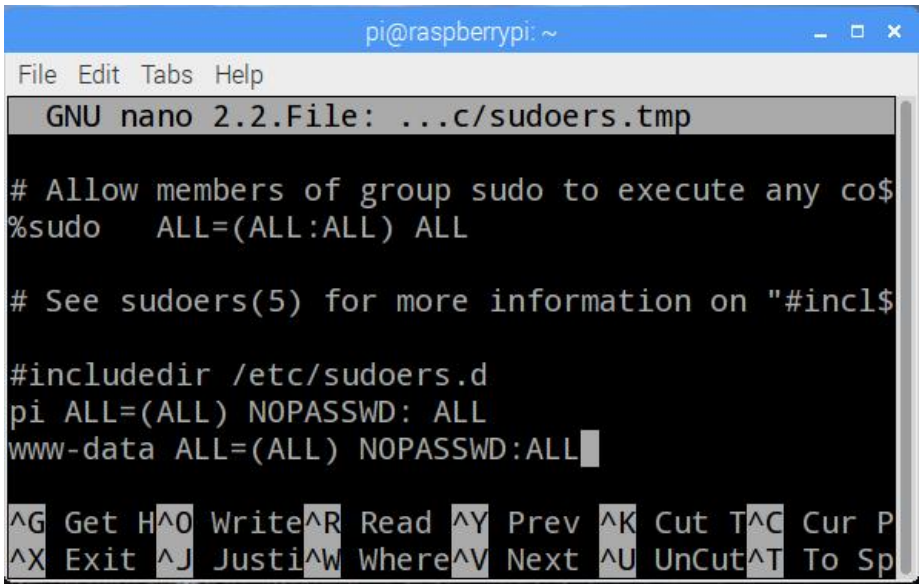

Figura 7. Fichero que da permisos de administrador a los usuarios.

Al terminar de hacer los pasos anteriores, la Raspberry Pi ya podría trabajar como servidor, y esto se comprobó ingresando la IP local de la Raspberry Pi en un navegador. Lo que se debería apreciar cuando se hace esto es la imagen de la figura 8. Para conocer la IP de la Raspberry pi se utiliza el comando `ifconfig`, como se puede ver en la figura 9.

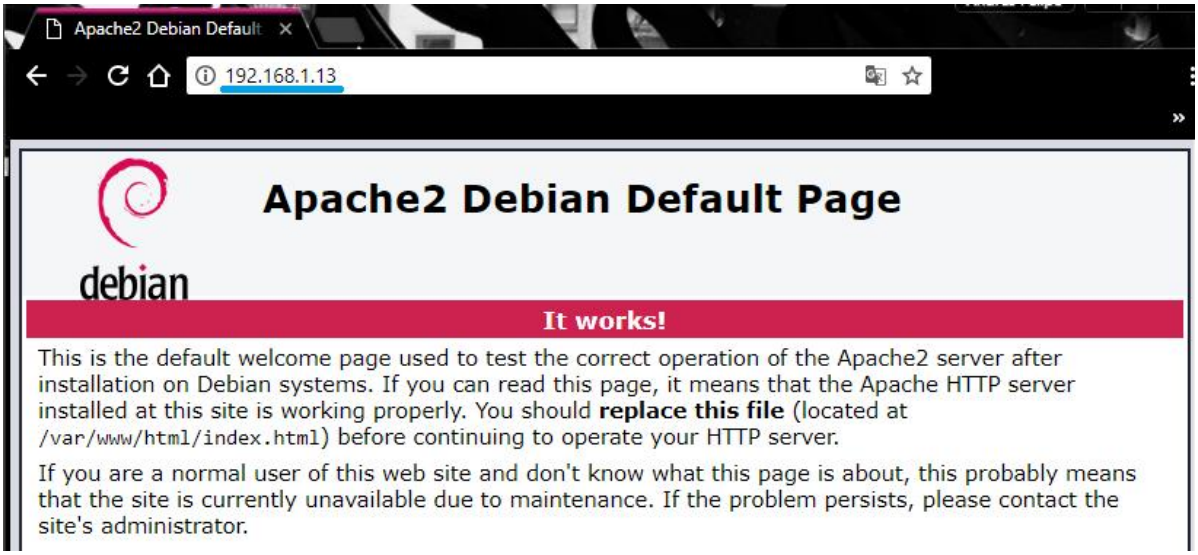

Figura 8. Servidor apache funcionando.

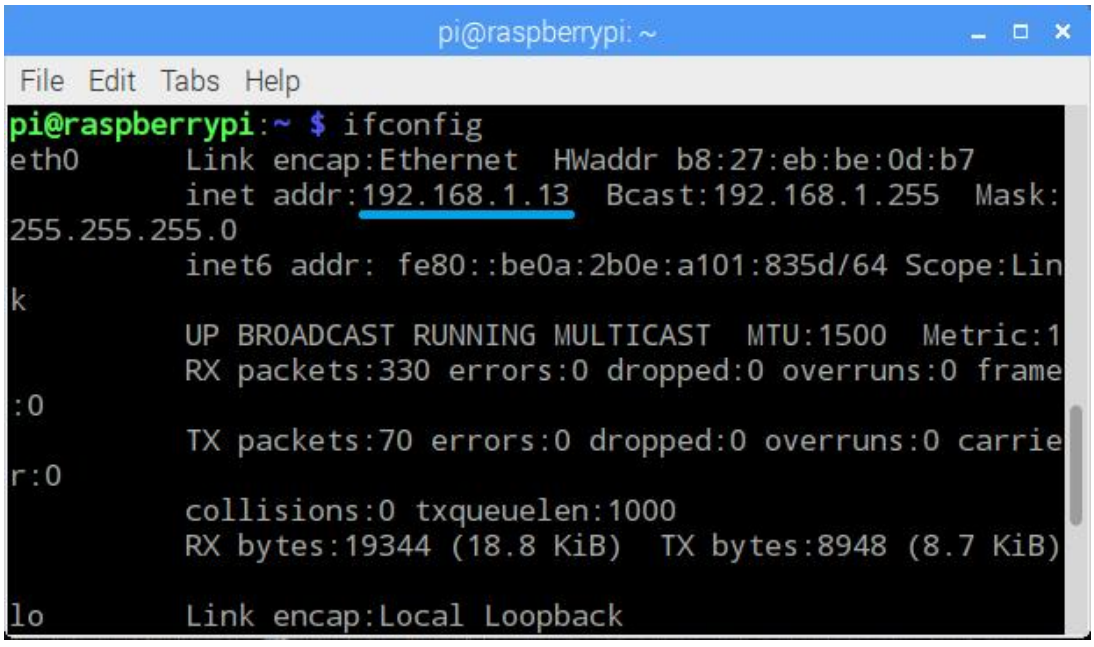

Figura 9. Ifconfig con la dirección IP de la Raspberry Pi.

6.1.2. Ingreso de scripts PHP y Python.

En esta sección se explica la correcta ubicación de los scripts de PHP y Python en la Raspberry para que puedan ser ejecutados correctamente. El script PHP es el

encargado de ejecutar los scripts de Python, que son los que controlan los pines del puerto GPIO.

El directorio var/www/html es el que usa apache, por eso este directorio debe contener todos los scripts PHP y Python. Primero se copiaron los scripts desde una computadora a una memoria USB y luego de la memoria USB al directorio var/www/html de la Raspberry Pi. Este es quizás el método más fácil para ingresar los scripts al directorio que ejecuta apache, pero se requiere que se esté trabajando en el entorno grafico de Debian. Para ingresar a este entorno se ingresa el comando *startx*.

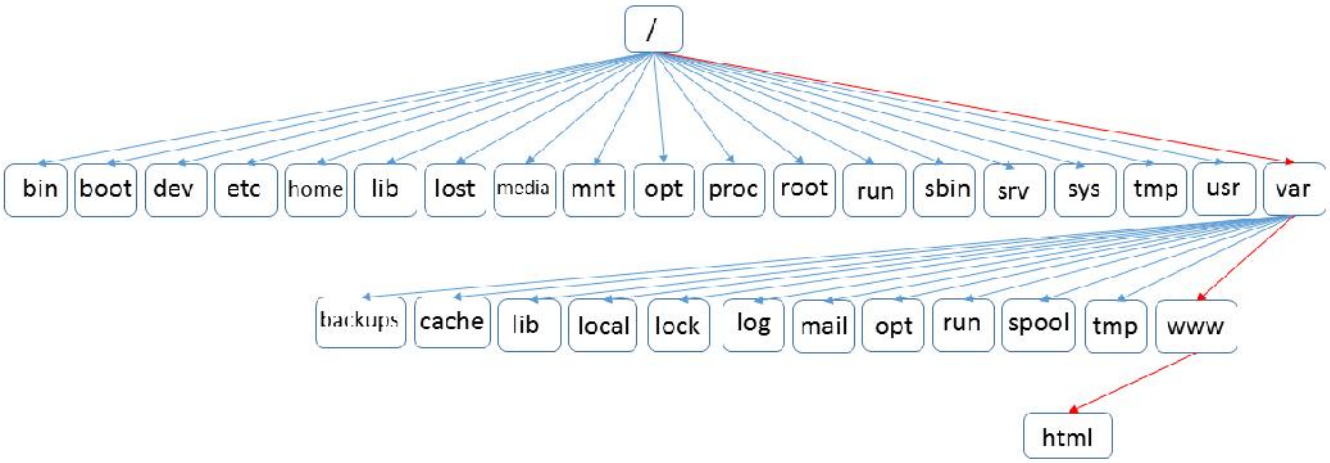

Figura 10. Ubicación de los scripts en el árbol de directorios.

6.1.3. Configuración de la computadora y la aplicación web.

A continuación, se explica cómo se configuró y usó el programa Wampserver para que el computador ejecutara la aplicación web sin problemas. También se explica el acondicionamiento que se le hizo al archivo index.php para que enviara la instrucción, vía internet, a la Raspberry Pi.

Debido a cambios en los protocolos de seguridad del navegador web Google Chrome, no se permite usar el micrófono en páginas que no sean seguras (http), pero si se permite para páginas que sí lo son (https). Esto implica que se tiene que ejecutar la aplicación web en un entorno que lo haga seguro. Esto se hace con la ayuda de la instalación de Wampserver. Esta aplicación se puede descargar fácilmente[3].

Se creó una carpeta dentro de la carpeta “www” y se le agregó todo lo que corresponde a la aplicación web. La carpeta “www” está en la dirección de

[3] http://www.wampserver.es/

instalación de Wampserver. Todos los elementos que constituyen la aplicación web pueden ser fácilmente descargados[4].

El archivo index.php es el encargado de recibir los comandos de voz, y por medio de librerías de Speech reconoce lo dicho y por el método POST lo envía a la Raspberry. Por lo que fue necesario verificar que la IP de envío que está en el archivo index.php sea igual a la de la Raspberry Pi, además de agregar la carpeta que lo contiene. las direcciones IP de las figuras 9 y 11 son iguales. Esto debe ser así para que la conexión entre la Raspberry Pi y la computadora sea exitosa.

```
function enviaAjax(event){

$.ajax({

  type: "POST",
  url: "http://192.168.1.13/procesa.php/",
  //direccion en que estan los scripts en la Raspberry pi
  data: {dato: event}, // se envia el dato al servidor por POST

});
```

Figura 11. Index mostrando dirección de ubicación del script PHP.

Para la utilización de la aplicación web fue necesario ejecutar Wampserver. Una vez se ejecutó el programa se ingresó localhost a la barra de direcciones del navegador. Automáticamente se abrió una página mostrando todos los elementos que contiene la carpeta de instalación de Wampserver. Se eligió la carpeta que contiene la aplicación web y este inmediatamente se ejecutó. A esta carpeta se le nombró artyom, por tanto, la dirección en el navegador para acceder a la aplicación web es localhost/artyom.

Se requirió de una investigación de los lenguajes de programación HTML, PHP y Python para el desarrollo de una nueva interfaz que permitiera el reconocimiento de la voz en modo continuo. Así el sistema de reconocimiento de voz es capaz de recibir muchos comandos de voz y transformarlos en una sola acción. Esto es útil cuando se quiere especificar en una orden la acción y el lugar. Por ejemplo: “encender habitación del primer piso”.

La aplicación web que se utilizó en este proyecto tiene algunas ventajas que la hacen una mejor opción que otras aplicaciones utilizadas en trabajos anteriores que usan un sistema de control por medio de la voz. Se puede configurar para que se narre un momento después la acción ordenada. Esto es de gran ayuda para personas que no puede corroborar visualmente lo ordenado. También tiene la posibilidad de

[4] https://sdkcarlos.github.io/sites/artyom.html

programar una acción para un conjunto de palabras. Esto permite que en una orden contenga el lugar u otras especificaciones de la acción.

6.2. Hardware.

6.2.1. Diseño de circuito electrónico.

Para este proyecto se usó una cantidad de 15 leds que simulan 15 bombillos de un hogar. Por la limitada cantidad de salidas GPIO que tiene la Raspberry, se tuvo que desarrollar un circuito electrónico que permitiera multiplicar las salidas, además de que permitiera encender más de un elemento a la vez, como ocurriría en una casa normal.

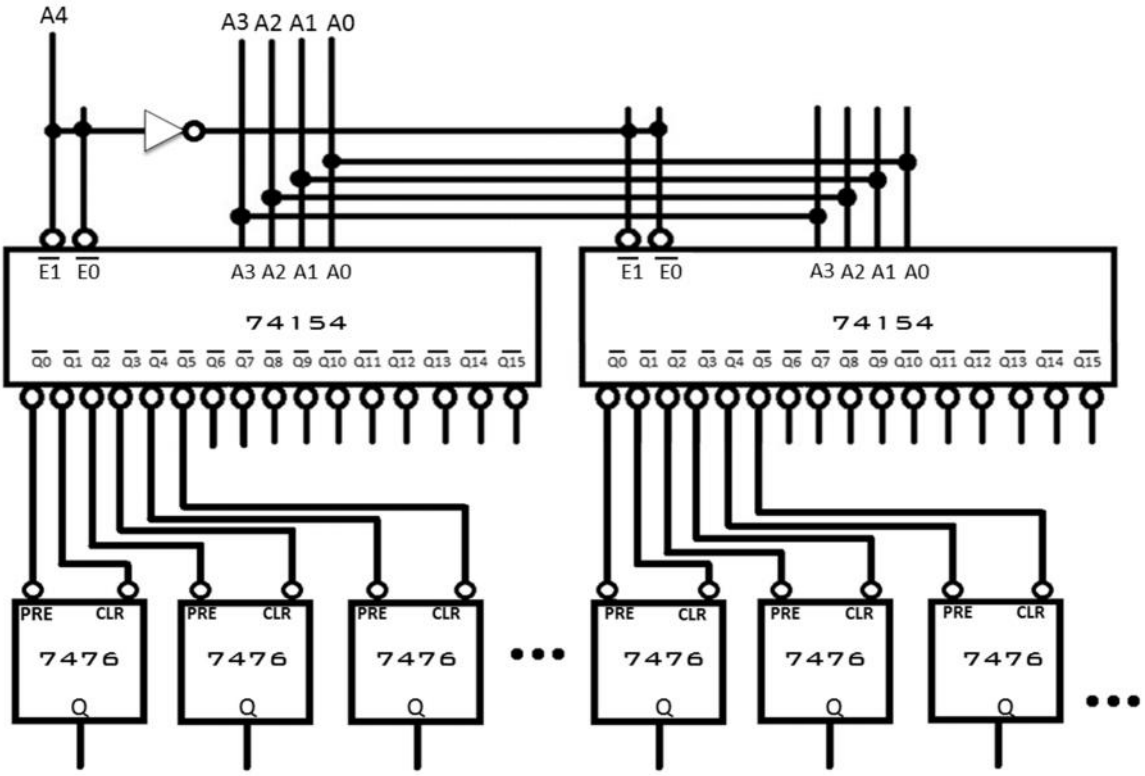

Figura 12. Diagrama circuito electrónico.

El diseño del circuito de la figura 12 contiene un decodificador, que tiene sus entradas conectadas a las salidas GPIO de la Raspberry, y sus salidas van conectadas a los terminales preset o clear de unos flip flop J-K. Estos flip flop se encargan de memorizar las acciones para que el decodificador quede libre para ejecutar otras acciones. Y por último, las salidas “Q” de los flip flop están conectadas a las resistencias que limitan el paso de corriente a los leds. Cada flip flop J-K se encarga de activar y desactivar un led, así que requiere de dos entradas, una de activación y otra de desactivación. La activación se logra con un “1” lógico en la entrada Preset

y la desactivación con un “1” lógico en la entrada Clear, como se puede ver en la tabla de verdad de la figura 3.

Al tener 15 leds, se requieren 15 flip flop, se necesita un decodificador de 5 a 32 y se necesitan 5 pines GPIO de la Raspberry. También se necesitan resistencias limitadoras de corrientes para proteger los leds. Y para calcularlas se utiliza como corriente máxima 30mA ya que este es el límite que soportan los leds.

$$V_{limitante} = \frac{V_{GPIO} - V_{led}}{I_{led}} = \frac{(5 - 0.7)V}{30mA} = 143.33\Omega$$

Conocido ya el valor para la Resistencia mínima, se escogió una Resistencia de mayor valor pero que permitiera un paso de corriente mayor a 5mA para que lograra encender el led, la cual fue de 620Ω.

$$I_{led} = \frac{V_{GPIO} - V_{led}}{R} = \frac{(5 - 0.7)V}{620\Omega} = 6.94mA$$

Por la naturaleza del comportamiento del flip flop J-K, es probable que algunos elementos del sistema estén encendidos, por lo tanto, se debió diseñar un script de Python que se asegure de apagarlos todos. Así se tienen 15 scripts de Python para encendido de leds, 15 para el apagado y 1 para apagar a todos.

Como las salidas del puerto GPIO de la Raspberry manejan tensiones menores a 3.3V y el circuito electrónico no trabaja con estas tensiones, fue necesario emplear un adaptador de nivel de 3.3V a 5V, el cual se puede ver en la figura 13.

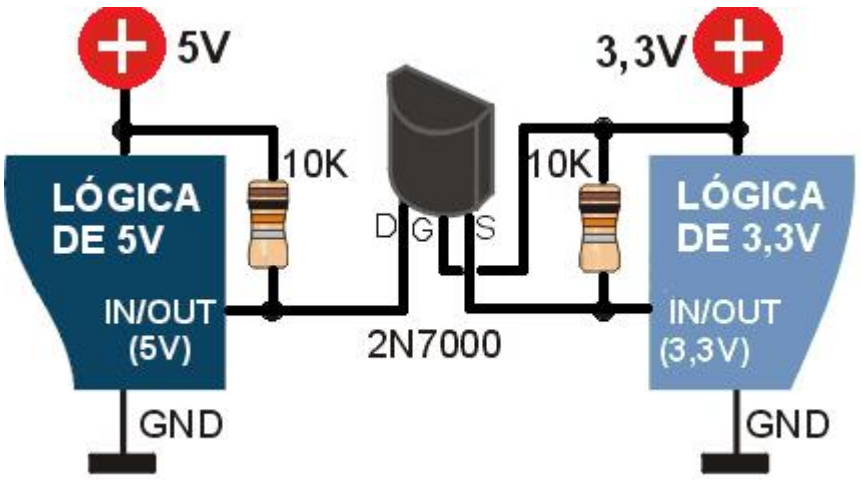

Figura 13. Adaptador de nivel de 3.3V a 5V.

7. Resultados

A continuación, se presentan los resultados obtenidos en las diferentes pruebas que se le hicieron al sistema domótico, en concordancia con los objetivos de este proyecto.

El correcto funcionamiento del reconocimiento hecho por Speech se comprobó de manera muy sencilla con ayuda de la aplicación web, la cual muestra en pantalla los comandos de voz reconocidos. De esta manera se comprobó que Speech en efecto reconociera todos los comandos de voz programados en la aplicación web, cada uno pronunciado 10 veces por lo menos por dos personas de diferente sexo. Las pruebas tan solo mostraron problemas con la palabra uno “1”, que en un 20% de las veces fue confundida con la palabra horno o con las palabras o no. Por esta razón es aconsejable tener especial cuidado con la pronunciación de esta palabra.

Después de esto se procedió a la segunda etapa de las pruebas, que constó de la comprobación del correcto funcionamiento del sistema cuando recibe el comando por parte de Speech hasta que el puerto GPIO activas sus pines a causa de este comando. Eso se hizo conectando los pines de puerto GPIO programados en los scripts de Python a cinco leds con sus resistencias limitadoras de corriente. Lo siguiente fue pronunciar las acciones programadas y ver si el código binario formado por los leds concordaba con el código asignado a ese comando. Este proceso lo hizo el sistema sin ningún tipo de problema. El montaje del circuito para esta prueba se puede apreciar en la figura 14.

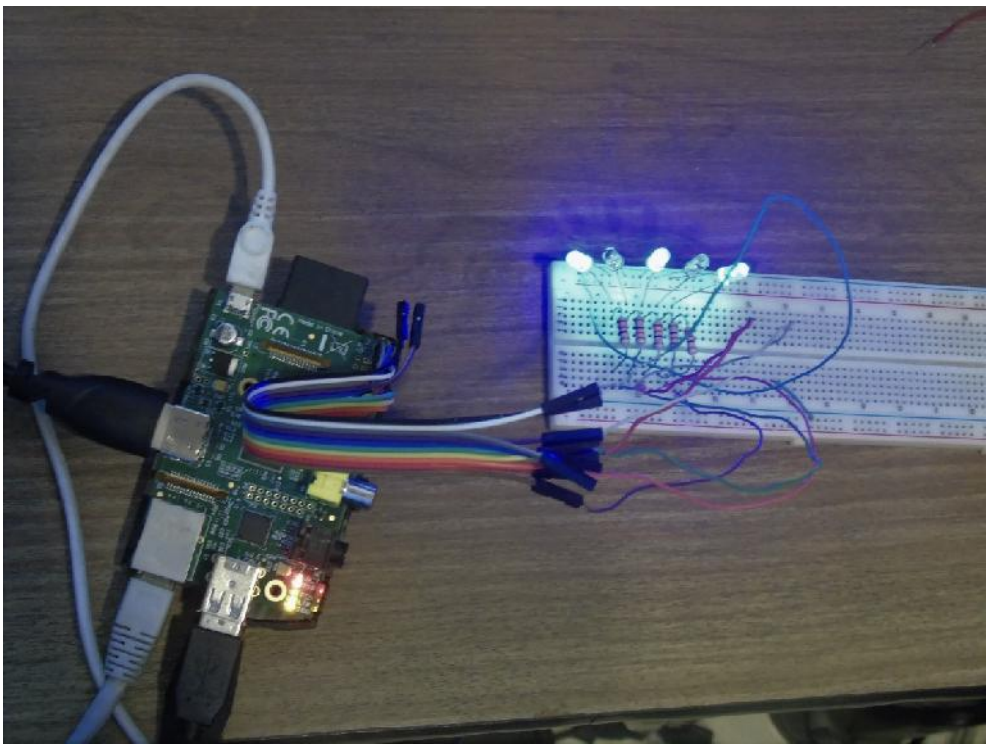

Figura 14. Circuito para la segunda etapa de pruebas.

El último paso de las pruebas fue comprobar si el circuito electrónico y el adaptador de nivel funcionaban juntos correctamente. Como con las pruebas anteriores se determinó una eficacia

alta del resto del sistema, se hicieron estas últimas pruebas con todo el sistema conectado. Una vez se puso a trabajar el sistema completo, se observó el comportamiento de los leds conforme se le daban las órdenes a la computadora. El montaje del circuito para esta prueba se puede apreciar en la figura 15.

Con estas pruebas se pudo comprobar el correcto funcionamiento del sistema de reconocimiento de voz controlando el entorno domótico simulado por el circuito electrónico.

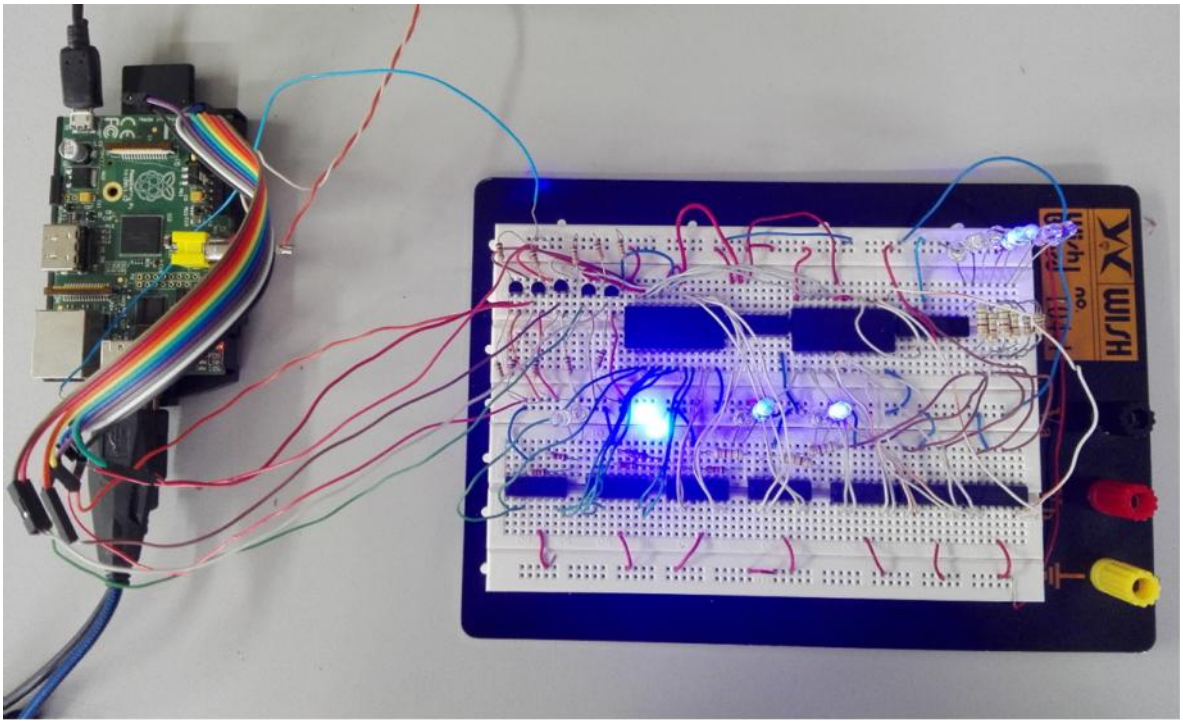

Figura 15. Sistema domótico simulado con leds.

8. Conclusiones

Con el desarrollo de este trabajo de grado se hace posible el uso de Raspberry Pi como herramienta de ejecución de un sistema de reconocimiento de voz con el objetivo de que reconozca ordenes de un usuario de manera remota usando protocolos de comunicación de red vía Wifi entre la Raspberry y un dispositivo compatible con el explorador Google Chrome y el software Wampserver. Para ejecutar las acciones programadas y posteriormente reconocidas se hace uso de los pines del puerto GPIO de la Raspberry Pi.

La utilización del Speech de Google en el proyecto significó una gran ayuda, lo cual permitió la simplificación del proyecto, ya que el entrenamiento de un sistema de reconocimiento de voz desde cero es un trabajo complejo. La integración del Speech de Google y la Raspberry Pi mostró resultados altamente satisfactorios, resultados que tienden a mejorar por la continua evolución que tiene este motor de reconocimiento.

En el desarrollo del proyecto se aprende funciones básicas de programación en los lenguajes Python, HTML y PHP, los cuales fueron de gran utilidad a la hora de programar la Raspberry y de la creación de la aplicación web utilizada como interfaz entre el usuario y la Raspberry Pi.

Con la utilización de decodificadores se puede elaborar un circuito que permita que la Raspberry PI controle un número de elementos igual a (2^n)/2, donde n es el número de salidas del puerto GPIO. Esto es, si se utilizan 17 pines del GPIO, entonces se puede hacer un circuito que permita controlar 65536 elementos.

9. Trabajo futuro

Trabajar con en una aplicación web que sea más versátil en cuanto a la recepción de comandos para que se pueda ejecutar una orden sin la necesidad que sea pronunciada en un orden o con las palabras exactas. Es decir, que el sistema ejecute de igual manera si se dice “encender bombilla sala” o “prender luz de la sala”.

Desarrollar una aplicación web que permita usar el micrófono en dispositivos Android sin tener problema con los protocolos de seguridad de Google. Que además permita ejecutar acciones que no solo tenga dos estados, como lo sería en control de temperatura o la apertura de una persiana.

10. Referencias

[1]. Raúl Diosdado, “Introducción a Raspberry Pi” [Online]. Available: https://www.zonamaker.com/images/contenido/raspberry/introduccion/descripcion_general/raspberry_descripcion.jpg

[2]. “Usando el puerto GPIO” [Online]. Available: http://diymakers.es/wp-content/uploads/2014/07/pins-RP.png

[3]. “About Debian” [Online]. Available: https://www.debian.org/intro/about

[4]. “Historia de la Domótica” [Online]. Available: http://www.arkiplus.com/historia-de-la-domotica

[5]. “Wampserver, Apache MYSQL y PHP en Windows” [Online]. Available: http://www.wampserver.es/

[6]. Amparo Varona Fernández, “Antecedentes y desarrollo de los sistemas actuales de reconocimiento automático del habla” [Online]. Available: http://hedatuz.euskomedia.org/6564/1/04321346.pdf

[7]. “Artyom.js HTML5 Voice Control” [Online]. Available: https://sdkcarlos.github.io/sitios/artyom.html

[8]. Jorge Leonardo Martínez Agudelo, “Diseño e implementación de un sistema de reconocimiento de voz mediante Raspberry Pi” [Online]. Available: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/7432/6213822M385d.pdf?sequence=1&isAllowed=y

[9]. “El decodificador” [Online]. Available: https://unicrom.com/decodificador/

[10]. “Adaptadores de nivel entre 5V y 3.3V” [Online]. Available: https://www.inventable.eu/2017/05/03/adaptadores-nivel-5v-3-3v/

[11]. Glen Shires, Google Inc. Hans Wennborg, Google Inc. “Web Speech API Specification” [Online]. Available: https://w3c.github.io/speech-api/speechapi.html

[12]. “Controles remotos con órdenes de voz” [online]. Available: http://colombia-inn.com.co/controles-remotos-con-ordenes-de-voz

[13]. “Speech recongnition HOWTO” [online]. Available: http://www.tldp.org/HOWTO/Speech-Recognition-HOWTO

[14]. “API Speech de cloud” [online]. Available: https://cloud.google.com/speech

A. Anexos

11.1 index.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title></title>
    <link rel="stylesheet" href="">

    <script src="artyom.min.js"></script>
    <script src="artyomCommands.js"></script>
    <script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>

    <style>
           .contenedor{
                  width: 60%;
                  margin-left: auto;
                  margin-right: auto;
           }
    </style>
</head>
<body>
    <div class="marquee-copy">
          <h1>Sistema Reconocimiento de Habla</h1>
          <h3>Para dar la orden: accion + lugar + enumeracion</h3>
             <h3>Ejemplo: encender habitacion 2</h3>


        </div>
    <div class="contenedor">

           <!-- botonera -->
           <div style="float: left;padding: 20px;">
                  <input type="button" onclick="startArtyom();" value="Activar
microfono">
                  <input type="button" onclick="stopArtyom();" value="Desactivarlo">
                  <input id="salida" style="width: 700px;" />
           </div>
    <div class="marquee-copy">
          <h2>comandos de voz habiles</h2>
          <h4>'encender Habitación 1','apagar habitación 1','encender habitación
2','apagar habitación 2','encender Habitación 3','apagar habitación 3',
                         'encender cocina','apagar cocina','encender baño 1','apagar
baño 1','encender baño 2','apagar baño 2','abrir persianas sala','cerrar persianas
sala',
                         'encender patio','apagar patio','encender sala','apagar
sala','encender comedor','apagar comedor',
```

```
                        'encender escaleras','apagar escaleras','encender
garaje','apagar garaje','abrir garaje','cerrar garaje','encender alarma','apagar
alarma',
                        'encender mirador','apagar mirador','apagar todo'</h4>
                <h2>otros comandos</h2>
                <h4>'limpiar','reiniciar','resetear'</h4>



        </div>
    </div>

    <script>
            //El sistema responde
            artyom.addCommands([
                    {
                        indexes:['encender Habitación 1','apagar habitación
1','encender habitación 2','apagar habitación 2','encender Habitación 3','apagar
habitación 3',
                                        'encender cocina','apagar cocina','encender
baño 1','apagar baño 1','encender baño 2','apagar baño 2','abrir persianas sala','cerrar
persianas sala',
                                        'encender patio','apagar patio','encender
sala','apagar sala','encender comedor','apagar comedor',
                                        'encender escaleras','apagar
escaleras','encender garaje','apagar garaje','abrir garaje','cerrar garaje','encender
alarma','apagar alarma',
                                        'encender mirador','apagar mirador','apagar
todo'],
                        action: function(i){
                                if (i==0) {
                                        artyom.say("Habitación 1 encendida");
                                enviaAjax(1)
                                }
                                if (i==1) {
                                        artyom.say("habitacion 1 apagada");
                                        enviaAjax(2)
                                }
                                if (i==2) {
                                        artyom.say("habitación 2 encendida");
                                        enviaAjax(3)
                                }
                                if (i==3) {
                                        artyom.say("habitación 2 apagada");
                                        enviaAjax(4)
                                }
                                if (i==4) {
                                        artyom.say("Habitación 3 encendida");
                                enviaAjax(5)
                                }
                                if (i==5) {
                                        artyom.say("Habitación 3 apagada");
                                enviaAjax(6)
                                }
                                if (i==6) {
                                        artyom.say("cocina encendida");
                                enviaAjax(7)
                                }
                                if (i==7) {
                                        artyom.say("cocina apagada");
                                enviaAjax(8)
                                }
                                if (i==8) {
                                        artyom.say("baño 1 encendido");
                                enviaAjax(9)
                                }
                                if (i==9) {
                                        artyom.say("baño 1 apagado");
                                enviaAjax(10)
                                }
                                if (i==10) {
```

```
          artyom.say("baño 2 encendido");
enviaAjax(11)
   }
   if (i==11) {
          artyom.say("baño 2 apagado");
enviaAjax(12)
   }
   if (i==12) {
          artyom.say("persianas sala abiertas");
enviaAjax(13)
   }
   if (i==13) {
          artyom.say("persianas sala cerradas");
enviaAjax(14)
   }
   if (i==14) {
          artyom.say("patio encendido");
          enviaAjax(15)
   }
   if (i==15) {
          artyom.say("patio apagado");
enviaAjax(16)
   }
   if (i==16) {
          artyom.say("sala encendida");
          enviaAjax(17)
   }
   if (i==17) {
          artyom.say("sala apagada");
enviaAjax(18)
   }
   if (i==18) {
          artyom.say("comedor encendido");
          enviaAjax(19)
   }
   if (i==19) {
          artyom.say("comedor apagado");
enviaAjax(20)
   }
   if (i==20) {
          artyom.say("escaleras encendidas");
          enviaAjax(21)
   }
   if (i==21) {
          artyom.say("escaleras apagadas");
enviaAjax(22)
   }
   if (i==22) {
          artyom.say("garaje encendido");
          enviaAjax(23)
   }
   if (i==23) {
          artyom.say("garaje apagado");
enviaAjax(24)
   }
   if (i==24) {
          artyom.say("garaje abierto");
          enviaAjax(25)
   }
   if (i==25) {
          artyom.say("garaje cerrado");
enviaAjax(26)
   }
   if (i==26) {
          artyom.say("alarma encendida");
          enviaAjax(27)
   }
   if (i==27) {
          artyom.say("alarma apagada");
enviaAjax(28)
   }
```

```
                                if (i==28) {
                                        artyom.say("mirador encendido");
                                        enviaAjax(29)
                                }
                                if (i==29) {
                                        artyom.say("mirador apagado");
                                enviaAjax(30)
                                }
                                if (i==30) {
                                        artyom.say("todo esta apagado");
                                enviaAjax(31)
                                }
                        }
                },
                {
                        indexes:['me voy','palo','culo','tetas','c***','m*****'],
                        action: function(){
                                alert('ok, groseria');
                        }
                },
                {
                        indexes:['limpiar','reiniciar','resetear'],
                        action: function(){
                                $('#salida').val('');
                        }
                }
        ]);

        // Escribir lo que escucha.
        artyom.redirectRecognizedTextOutput(function(text,isFinal){
                var texto = $('#salida');
                if (isFinal) {
                        texto.val(text);
                }else{

                }
        });


        //inicializamos la libreria Artyom
        function startArtyom(){
                artyom.initialize({
                        lang: "es-ES",
                        continuous:true,// Reconoce 1 solo comando y para de escuchar
             listen:true, // Iniciar !
             debug:true, // Muestra un informe en la consola
             speed:1 // Habla normalmente
                });
        };

        // Stop libreria;
        function stopArtyom(){
                artyom.fatality();// Detener cualquier instancia previa
        };

    function enviaAjax(event){

    $.ajax({

      type: "POST",
      url: "http://192.168.0.10/scripts/procesa.php/",
         //direccion en que estan los scripts en la Raspberry pi
      data: {dato: event}, // se envia el dato al servidor por POST

    });
  }

  // });
  </script>
</body>
</html>
```

11.2. Procesa.php

```
<?php

$valor = $_POST['dato'];
 // recibir el dato del navegador por metodo POST

if($valor=='1')
{
    exec('sudo python 1.py');
    $retorna=1;
}
if($valor=='2')
{
    exec('sudo python 2.py');
    $retorna=2;
}
if($valor=='3')
{
    exec('sudo python 3.py');
    $retorna=3;
}
if($valor=='4')
{
    exec('sudo python 4.py');
    $retorna=4;
}
if($valor=='5')
{
    exec('sudo python 5.py');
    $retorna=5;
}
if($valor=='6')
{
    exec('sudo python 6.py');
    $retorna=6;
}
if($valor=='7')
{
    exec('sudo python 7.py');
    $retorna=7;
}
if($valor=='8')
{
    exec('sudo python 8.py');
    $retorna=8;
}
if($valor=='9')
{
    exec('sudo python 9.py');
    $retorna=9;
}
if($valor=='10')
{
    exec('sudo python 10.py');
    $retorna=10;
}
if($valor=='11')
{
    exec('sudo python 11.py');
    $retorna=11;
}
if($valor=='12')
```

```
{
    exec('sudo python 12.py');
    $retorna=12;
}
if($valor=='13')
{
    exec('sudo python 13.py');
    $retorna=13;
}
if($valor=='14')
{
    exec('sudo python 14.py');
    $retorna=14;
}
if($valor=='15')
{
    exec('sudo python 15.py');
    $retorna=15;
}
if($valor=='16')
{
    exec('sudo python 16.py');
    $retorna=16;
}
if($valor=='17')
{
    exec('sudo python 17.py');
    $retorna=17;
}
if($valor=='18')
{
    exec('sudo python 18.py');
    $retorna=18;
}
if($valor=='19')
{
    exec('sudo python 19.py');
    $retorna=19;
}
if($valor=='20')
{
    exec('sudo python 20.py');
    $retorna=20;
}
if($valor=='21')
{
    exec('sudo python 21.py');
    $retorna=21;
}
if($valor=='22')
{
    exec('sudo python 22.py');
    $retorna=22;
}
if($valor=='23')
{
    exec('sudo python 23.py');
    $retorna=23;
}
if($valor=='24')
{
    exec('sudo python 24.py');
```

```
    $retorna=24;
}
if($valor=='25')
{
    exec('sudo python 25.py');
    $retorna=25;
}
if($valor=='26')
{
    exec('sudo python 26.py');
    $retorna=26;
}
if($valor=='27')
{
    exec('sudo python 27.py');
    $retorna=27;
}
if($valor=='28')
{
    exec('sudo python 28.py');
    $retorna=28;
}
if($valor=='29')
{
    exec('sudo python 29.py');
    $retorna=29;
}
if($valor=='30')
{
    exec('sudo python 30.py');
    $retorna=30;
}
if($valor=='31')
{
    exec('sudo python 31.py');
    $retorna=31;
}
echo $retorna;
?>
```

11.3. 1.py

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.cleanup()

GPIO.setup(4, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.HIGH)
time.sleep(0.5)
```

```
GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.1)
```

11.4 31.py

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.cleanup()

GPIO.setup(4, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
```

```
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.HIGH)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.HIGH)
GPIO.output(17, GPIO.HIGH)
GPIO.output(27, GPIO.HIGH)
GPIO.output(22, GPIO.HIGH)
```

```
GPIO.output(23, GPIO.LOW)
time.sleep(0.5)

GPIO.output(4, GPIO.LOW)
GPIO.output(17, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
GPIO.output(22, GPIO.LOW)
GPIO.output(23, GPIO.LOW)
time.sleep(0.1)
```